PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 17/30, 9/45 // 3/033

A1

(11) International Publication Number: WO 99/04349

(43) International Publication Date: 28 January 1999 (28.01.99)

(21) International Application Number: PCT/IB98/00697

(22) International Filing Date: 11 May 1998 (11.05.98)

(30) Priority Data:
97202193.5 15 July 1997 (15.07.97) EP
(34) Countries for which the regional or international application was filed: NL et al.

(71) Applicant: KONINKLIJKE PHILIPS ELECTRONICS N.V. [NL/NL]; Groenewoudseweg 1, NL-5621 BA Eindhoven (NL).

(71) Applicant (*for SE only*): PHILIPS AB [SE/SE]; Kottbygatan 7, Kista, S-164 85 Stockholm (SE).

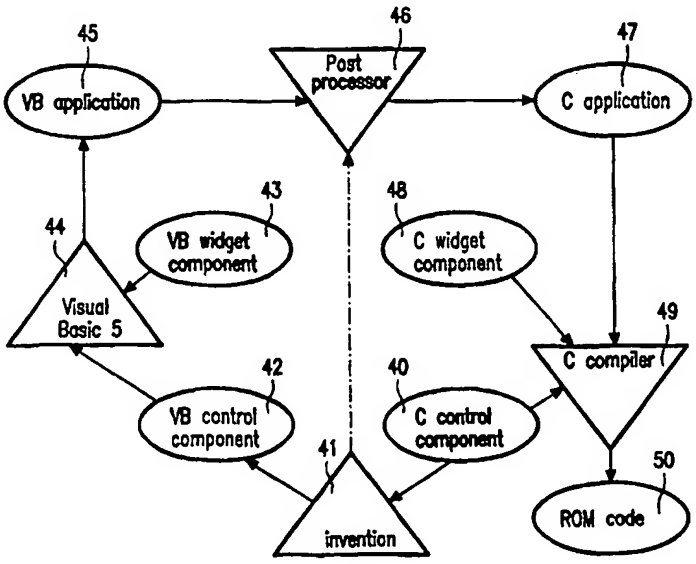
(72) Inventors: BLOEM, Gerrit-Jan; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL). MORSELT, Frank, Anton; Prof. Holstlaan 6, NL-5656 AA Eindhoven (NL).

(74) Agent: GROENENDAAL, Antonius, W., M.; Internationaal Octrooibureau B.V., P.O. Box 220, NL-5600 AE Eindhoven (NL).

(81) Designated States: JP, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: A METHOD AND SYSTEM FOR DESIGNING A GRAPHICAL USER INTERFACE FOR AN ELECTRONIC CONSUMER PRODUCT



(57) Abstract

A method and system for effectively and efficiently constructing a development environment for graphical user interfaces for an electronic consumer product. The basic aim of the invention tool is to automatically generate an interface in the authoring environment for customer specific target control components. This is done so by automatically creating a host platform control component (in the format of an OCX) from an existing target platform control component (in the format of a C header file). After running the invention once for every control component, a designer can then write on the host, for instance Visual Basic application code that performs calls to properties and methods of these generated control components.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

A method and system for designing a graphical user interface for an electronic consumer product.

BACKGROUND OF THE INVENTION

The invention relates to a method according to the preamble of Claim 1. Present-day consumer products, such as for example television sets, and the like, get implemented with many user features, that must individually as well as collectively be rendered accessible to a user person through an appropriate graphical user interface. Now, a graphical user interface in the first place contains graphical objects that may include items like buttons, sliders, hotspots, and indicators of various kinds such as icons and text. Further elements of such interface are functional objects that link the graphical objects to an associated underlying functionality. The graphical objects are represented by so-called **widget components** that contain the facilities and restrictions of the underlying display hardware. An example of such widget may have *properties* like "backgroundcolour", *events* like "clicked" and *methods* like "redraw". The functional objects are represented by so-called **control components**. An example is a tuner component, having *properties* like "maximumfrequency", *events* like "channelfound", and *methods* like "setchannelnumber".

Now for constructing a graphical user interface in an effective and efficient manner, it is necessary to have available a well-defined design environment that provides a true likeness (WYSIWYG) of the eventual imagery on the display of the product. To build such an environment for a particular graphic IC is expensive, and moreover, may be targeted to an uncertain realization. With respect to the target platform, the defining of features of the display hardware is nowadays realized through embedded and callable library software. Such libraries may in time be frequently amended in successive versions of the display hardware in question or in replacement types for that display.

To be able to define the coupling between the two categories of component, the designer of an application should know the application programming interface (API) of the control components. This interface could be defined by various different economic entities in the manufacturing column and may not be uniform in time. Both widget components and control components on the target platform are generally defined in a general purpose language like C that has a relatively low degree of abstraction.

Therefore, implementation of graphical user interfaces directly on the

target is time-consuming and costly. Now there exist generic platforms like the PC, that allow the running of various relatively higher abstracted programming environments such as Microsoft Visual Basic that are well suited for creating graphical user interfaces on that platform. Nowadays more and more of those generic development environments are used for constructing graphical user interfaces on specific hardware. In that case communication to the host platform (the PC) of the facilities and restrictions of the target platform is crucial.

SUMMARY TO THE INVENTION

In consequence, amongst other things, it is an object of the present invention to allow the automatic defining of all kinds of control components in the graphical user interface as based on the underlying library software, while recognizing that widget components on a host platform such as a PC, and a target system, such as a television set, may be too wide apart from each other to be convertible automatically.

Therefore, in its most extensive version, the following aspects are necessary, although in less complete solutions, a subset of the following may suffice:

- Widget components on the host platform representing the widget components of the target, so that the application programmer (designer) will have the facilities and restrictions of the target platform available.
- Interfaces of the control components on the host platform representing the control components of the target, so that the application programmer may indicate the link from the graphics to the underlying functionality. Optionally, the application programmer may construct the implementation of the interface, thereby effecting a certain degree of functionality simulation on the host.
- A compiler that translates the host application to a target application.

Now therefore, according to one of its aspects the invention is characterized by the items recited in the characterizing part of Claim 1.

The invention also relates to a system being arranged for implementing the above method. Further advantageous aspects of the invention are recited in dependent Claims.

BRIEF DESCRIPTION OF THE DRAWING

The invention provides a tool to generate the framework of a control OCX (cf. the glossary) from a C library of target control code. In a particular embodiment, the invention should be an integral part of a graphical development environment on a generic

platform, such as Visual Basic for the PC. The generated control OCX will generally not perform the control functions, inasmuch as it will be decoupled from the hardware. However it will present the same interface inside the authoring environment as available on the target platform, and thus avoid mismatch after subsequent code generation. Such and further aspects and advantages of the invention will be discussed more in detail along with the disclosure of preferred embodiments hereinafter, and in particular with reference to the appended Figures that show:

Figure 1, general architecture of a compiler system;

Figure 2, overview of an authoring environment;

Figure 3, positioning of the invention in the architecture of an authoring environment;

Figure 4, an exemplary part of the input for the invented compiler;

Figure 5, an exemplary part of the use of the output of the invented compiler in an application designed on the host.

VOCABULARY

For better insight into the invention, the following definitions are given:

API

Application Programming Interface

Component

A configurable part of the system which has properties, methods and responds to events. Within Visual Basic components are OCX-es.

Container

An OLE container which may contain OCX-es.

Control component

A component that controls part of the functionality of a TV or other display.

Event

A trigger from a user input device or system device, e.g. remote control 'volume up', front panel 'off', channel tuned signal.

Method

A function belonging to an OCX that can be called at runtime by a container application like Visual Basic.

OCX

OLE Control Extension. A self contained plug in, binary software module that can only be used inside an application like Visual Basic. For example, a button or a timer.

OLE

Object Linking and Embedding: Microsoft's component based software technology.

Post Processor

Name commonly used for the target code generator.

Property

A variable belonging to an OCX that can be changed at either design or run time by a container application such as Visual Basic, e.g. a button's colour or icon, a timer's timeout.

Target System

The graphic IC for which authoring is required.

Target widget

A (graphical) component in a form suitable for use on the target system.

Visual Basic

Graphical development environment for the creation of Microsoft Windows applications. Visual Basic may inter alia be used to create OCX-es.

Widget

Window gadget, a component with a representation on screen, e.g. button, slider.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Figure 1 shows the general architecture of a compiler system. Item 20 represents the input code that comes as a finite-length string of formatted information items defined according to an appropriate origin syntactic structure. For compiling, the string is first fed to parser subsystem 22, that under execution of various analyzing substeps, divides the string in functional code units, whilst maintaining any pre-existing cross-referencing among the various functional code units as received. This parsing then will create an intermediate data structure symbolized by item 24 that usually is constructed around some notion of hierarchy, and has thus been symbolized by a pyramid-like shape. In subsystem 26, the structure so formed is processed under the constraints of a chosen target syntactic structure, to thereby produce an output code 28 that thereby constitutes a mapping of the original code functionality in another language. By itself, compilers are tools that allow automatic conversion of virtually any program into another computer language, and as a general class, compilers represent common general knowledge.

Figure 2 gives an overview of the authoring environment from a user's point of view. The **host** of the authoring system is a PC; the **target** is a consumer product (for example a TV-set or handheld telephone) which contains one or more ICs for graphic (OSD) and control functionality. Widget and control components are provided, both on the host side as well as on the target side, having the same functionality on the host and target side. The components on the target side are realized as C software libraries. The components on the host side are realized as OCX-es. An application designer uses the supplied host components in order to create and test an application at a high abstraction level. Once the designer is satisfied with a design after simulation on the PC, this design can be processed (translated) automatically to target software, which will then be compiled and linked with the target components. Finally the application can be tested on the target.

Figure 3 shows the invention (block 41) that transforms a Control API in the form of a C header file (block 40) into a Control OCX (block 42), and some other subsystems of the authoring environment. The figure does not cover the total authoring environment system but only those subsystems that are relevant to the invention. Block 44 represents Visual Basic version 5 that forms the base of the authoring environment. In Visual Basic the designer can use two types of OCX'es to create an application: the Widget OCX'es (block 43) for the graphical part of the user interface of the application and the Control OCX'es (block 42) to control the hardware (for example the TV). Block 46 represents the Post Processor, a compiler that translates the application (block 45) that is designed in Visual

Basic into a C application (block 47). The Post Processor creates this application by translating all instances of Widget and Control OCX'es with their properties and Visual Basic code to C code. Finally a general C compiler (block 49) may be used to compile the C program to ROM code (block 50).

Figure 4 illustrates an exemplary part of the compiler input. The input is a C header file containing variables and function declarations (block 48 of figure 2). The header file specifies how to set or read the state of a piece of hardware by a C program. In this example there exists a header file that specifies how to control a tuner. For example, you can set the frequency range, used by the tuner when it is searching for valid programs, by setting the minfreq and maxfreq variables to a certain value, and assigning a channel number to a frequency by calling the SetChannelNumber function.

Figures 5a, 5b, in correspondence with Figure 4, illustrate an example of how to use the output of the invented compiler (block 41 in figure 3) in the authoring environment. The output (the generated Control OCX, block 42 in figure 3) is the white square with the name of the header file on it ("tuner" in this case). You can use this object in Visual Basic at design-time in the same way you use the objects to construct a graphical user interface (the widget OCX'es, block 43 in figure 3). During the design, the location, orientation, size, shape, caption, name, colour and various other attributes of a particular control may be amended, by keying, dragging and other control mechanisms.

In Figure 5a, the tuner Control OCX is placed together with some Widget OCX'es representing the graphical user interface of the application. Since the variable declarations of the tuner header file are translated to properties of the tuner Control OCX and the function declarations of the tuner header file are translated to methods of the tuner Control OCX, it is possible to simulate the manipulation of the concerned piece of hardware (the tuner in this example).

In Figure 5b this is illustrated by a piece of code that is written by the application designer in the code window of Visual Basic. In this particular case, the SetChannelNumber function will be called if the 'Button1 control' sends the event that it has been pressed with the OK-key of the remote control. Due to the fact that the tuner is not part of the graphical part of the designed application, it will be invisible when the application is executed in Visual Basic run-time mode. Further, according to the Visual Basic syntax, various widget components have been specified in the Figure, that respectively represents the tuner menu, the set definition, the specifying of a quantity, here the channel number "12", the incrementing and decrementing buttons, and one control component, the newly

introduced "tuner" component.

The invention produces a Visual Basic control file (.ctl) and project file (.vbp) which together form a Visual Basic project and can be compiled into a control OCX or used as a Visual Basic subproject.

The invention translates C code constructs into higher level programming language constructs. In general, such higher level language has less detailed constructs available, for instance C is strongly typed and allows for pointer arithmetic, while Visual Basic language is/does not. Basically, the translation from C to Visual Basic is a non-injective mapping function and information is lost. The invention should provide this missing information to the Post Processor that will re-generate this information. One possible solution is to let the invention generate a conversion table that the Post Processor can use.

When the invention has generated the control OCX, it will become available for the designer in the standard Visual Basic toolbox indicated by a standard icon. All control components are indicated by the same icon, however they can be distinguished by moving the cursor on top of the icon and reading the name of the component. Every control component bears the same name as its original C header file from which it was created. When an instance of such a control component is placed on a VB form, it is visualized on this form, for instance by a white rectangle showing its name in the left top corner (see Figure 5a).

In simulation mode, the application code can read/write properties, call methods, and react on events from the generated control component. Initially, these actions have no effect, since the bodies on the methods in the generated components are empty (they are just *stubs*). However, it is possible to manually write Visual Basic code in the bodies so that it has some functionality in simulation mode. An example of such functionality could be a simulation panel that shows all current (emulated) hardware parameter values. This way, the designer will get some feedback on the control components in his design. Obviously, you can not test the control functionality itself, because it is decoupled from the target hardware.

The method described herein may to a large part be applicable to another type of application than graphical user interfaces. The principle is that a part of the features translates immediately from host to target, whereas for others, the host has only dummy functionality, whereas the target plaform is fully operative. The automatic compilation again ensures the immediate transferability.

CLAIMS:

1. A method for constructing on a host platform a graphical development environment for the creation of an application program on the target platform, generally in a consumer product, the development environment comprising an arrangement of graphical components and control components, each defined in terms of a associated parameter set, offering an equivalent functionality on the host platform as is available on the target platform,

said method being characterized by fully equivalent graphical components on both host and target and control components that on the host have similar interfaces as on the target, where this latter similarity is guaranteed by an automatic compilation of the target control component interface to the control components interface on the host, while at the same time additional information is generated that allows a unique and correct implementation from the host to the target of the designed user interface.

2. A method as claimed in Claim 1, whilst providing said host-specific manner in a visual design environment.

3. A method as claimed in Claim 1 or 2, whilst using a C header file for the interface of the control components in the software library.

4. A method as claimed in Claims 1, 2 or 3, wherein said environment is Visual Basic.

5. A method as claimed in Claims 1 to 4, for use with respect to a screen based product.

6. A method as claimed in any of Claims 1 to 5, being applicable to another type of applications than graphical user interfaces.

7. A system being arranged for implementing a method as claimed in any of Claims 1 to 6.

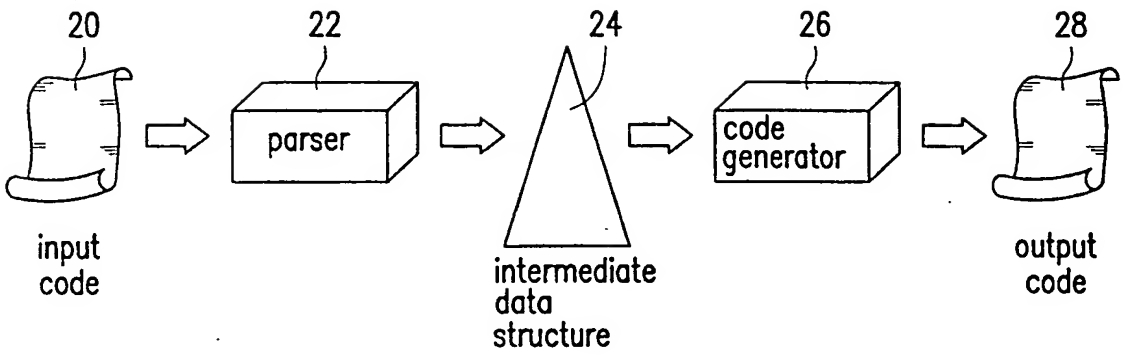


FIG. 1

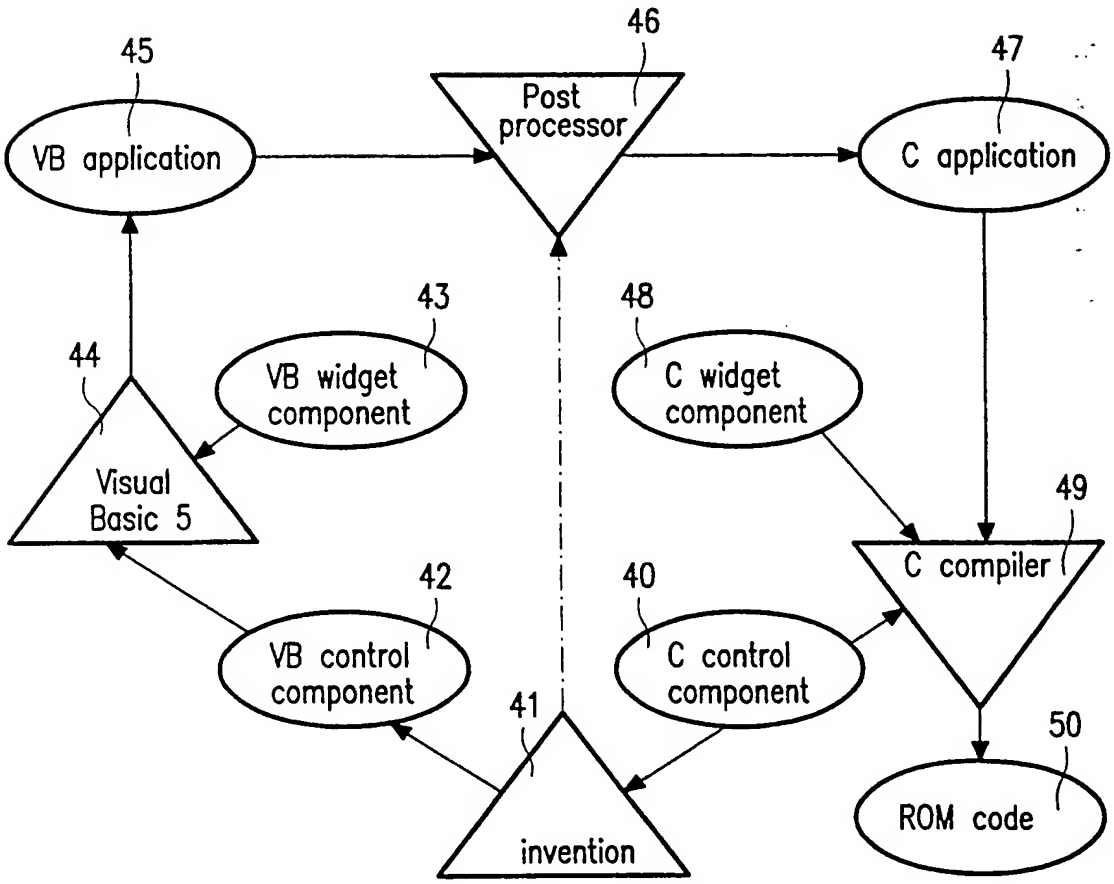


FIG. 3

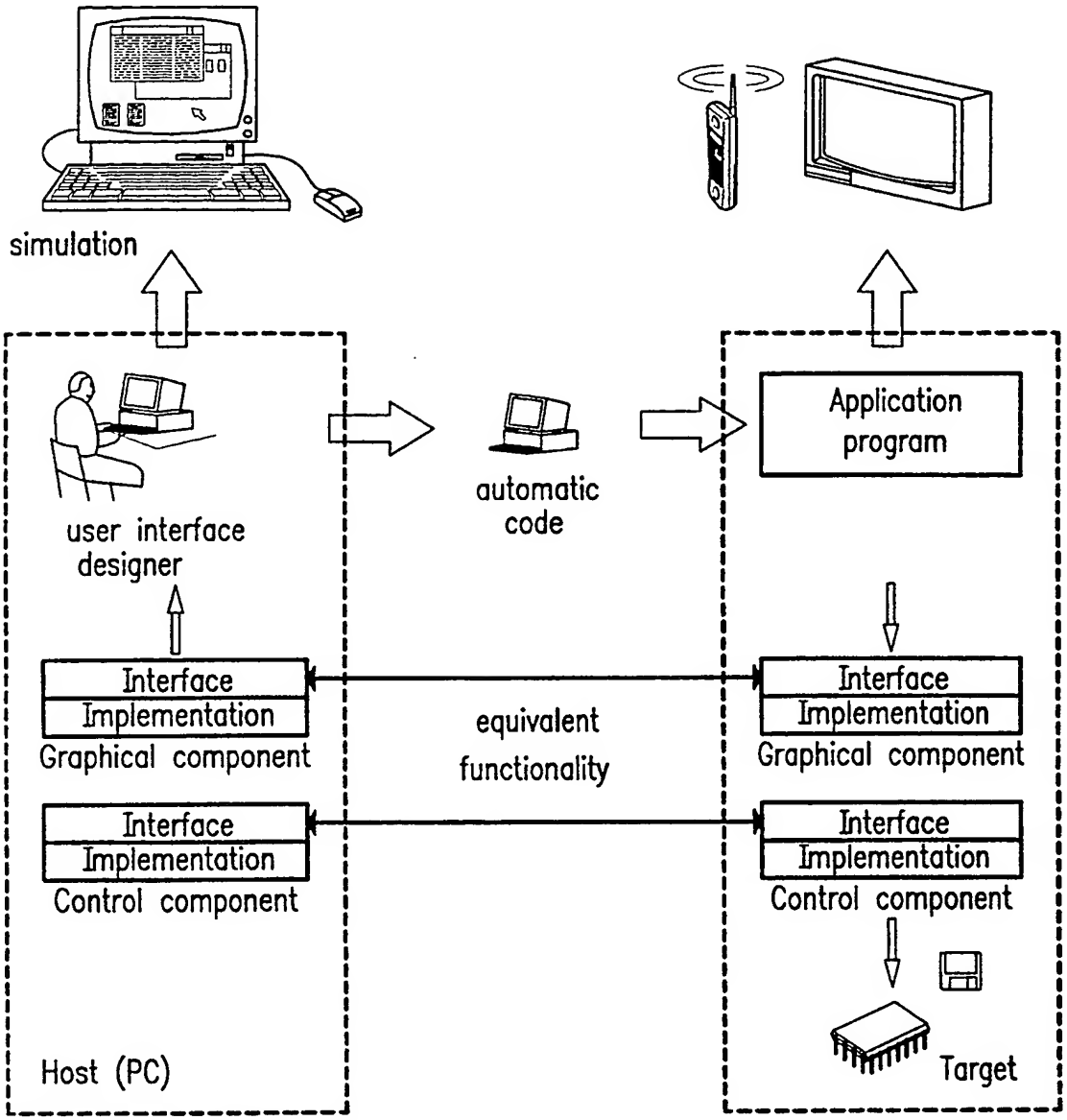


FIG. 2

```
float minfreq, maxfreq;
void SetChannelNumber(int channel, float freq);
float GetChannelFreq(int channel);
char* GetChannelName(int channel);
```

FIG. 4

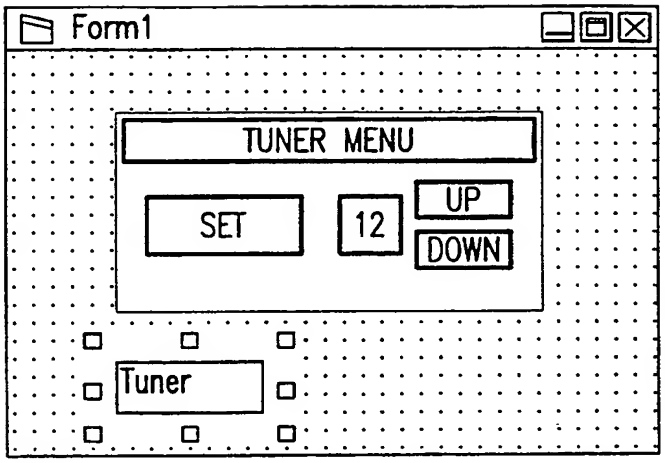


FIG. 5a

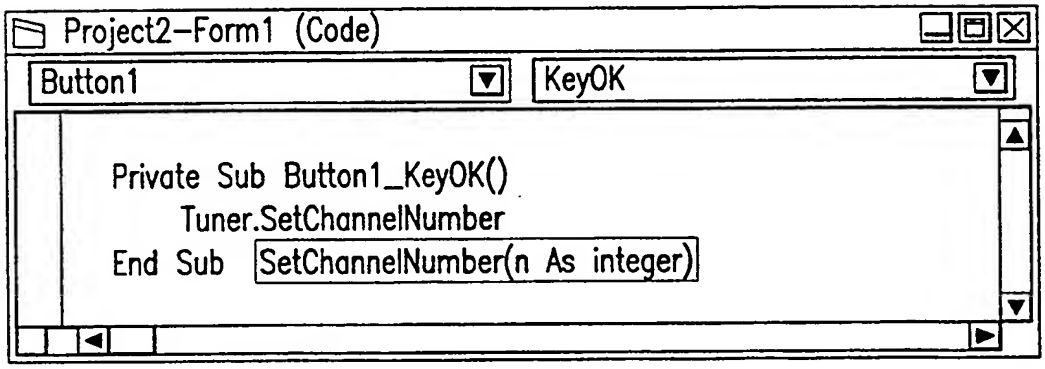


FIG. 5b

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB 98/00697

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G06F 17/30, G06F 9/45 // G06F 3/033
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPIL, DIALOG, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5404441 A (MOTOAKI SATOYAMA), 4 April 1995 (04.04.95), column 1, claims 1-2, abstract --	1-7
Y	Choosing a User Interface Development Tool, Volume 14, No 4, 1997, Laura A. Valaer et al, "Choosing a User Interface Development Tool", see the whole document --	1-7
A	PC Week, Volume 13, No 11, March 1996, Peter Coffee, "Windows gets a crack at Taligent", see the whole document --	1-7

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" erlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 December 1998	28 -12- 1998
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Linus Wretblad Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB 98/00697

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5490245 A (THEODORE D. WUGOFSKI), 6 February 1996 (06.02.96), column 1 - column 2, claims 1,16,22,28, abstract --	1-7
A	Patent Abstracts of Japan, abstract of JP 8-278881 A (TOSHIBA KK), 22 October 1996 (22.10.96) --	1-7
A	Patent Abstracts of Japan, abstract of JP 8-166874 A (MEIDENSHA CORP), 25 June 1996 (25.06.96) --	1-7
A	Patent Abstracts of Japan, abstract of JP 6-230953 A (MITSUBISHI ELECTRIC CORP), 19 August 1994 (19.08.94) --	1-7

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter[illegible]nal application No.
PCT/IB 98/00697

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5404441 A	04/04/95	JP 6004277 A	14/01/94
US 5490245 A	06/02/96	US 5559947 A	24/09/96